Amendments to the Specification

Please replace the paragraph that begins on Page 1, line 6 and carries over to Page 2, line 4 with the following marked-up replacement paragraph:

-- The present invention is related to U. S. Patent _____ (serial number 09/669,227, filed 09/25/2000), titled "Object Model and Framework for Installation of Software Packages Using Instantiated Objects"; U. S. Patent ~~JavaBeans™"; U. S. Patent~~ _____ (serial number 09/707,656, filed 11/07/2000), titled "Object Model and Framework for Installation of Software Packages Using Object Descriptors"; U. S. Patent _____ (serial number 09/707,545, filed 11/07/2000), titled "Object Model and Framework for Installation of Software Packages Using Object REXX"; U. S. Patent _____ (serial number 09/707,700, filed 11/07/2000), titled "Object Model and Framework for Installation of Software Packages Using Structured Documents"; and U. S. Patent _____ (serial number 09/879,694, filed 06/12/2001), titled "Efficient Installation of Software Packages". These inventions are commonly assigned to the International Business Machines Corporation ("IBM") and are hereby incorporated herein by reference. --

Please replace the paragraph that begins on Page 6, line 18 and carries over to Page 7, line 3 with the following marked-up replacement paragraph:

-- The related inventions teach use of an object model and framework for software installation packages and address many of these problems of the prior art, enabling the installation process to be simplified for software installers as well as for the software developers who must prepare their software for an efficient, trouble-free installation, and define several

techniques for improving installation of software packages. While the techniques disclosed in the related inventions provide a number of advantages and are functionally sufficient, there may be some situations in which the techniques disclosed therein may be improved upon. --

Please replace the paragraph that begins on Page 12, line 20 and carries over to Page 14, line 4 with the following marked-up replacement paragraph:

-- A distributed directory facility, which is referred to hereinafter as a Lightweight Directory Access Protocol ("LDAP") directory or directory server for ease of reference, may be installed on one or more devices in the network environment of Fig. 2. For example, application server 47 may include an LDAP directory. Or, application server 47 may access another device on which an LDAP directory is installed. Data repositories used by the LDAP directory may reside at one or more locations within the environment. (Note that the term "distributed directory" is used herein for purposes of illustration and not of limitation: the present invention may be used with directory implementations which do not span more than one device.) Furthermore, LDAP directory facilities may be available on end-user devices such as device 10. An example of this latter case is the Windows® 2000 operating system from Microsoft Corporation, which uses LDAP directory facilities for its "Active Directory". Commercial LDAP directory implementations are widely available, and are well known in the art. A detailed description of such implementations is therefore not deemed necessary for purposes of the present invention. Preferred embodiments of the present invention may use any such LDAP directory implementation which supports basic functions of (1) data storage and retrieval based upon defined access rights or privileges and (2) authentication of requesters. (Alternative

embodiments do not require authentication of requesters, as will be described below with reference to Fig. 10.) Note that while preferred embodiments [[use]] use an LDAP directory facility that may be distributed across multiple devices, an implementation which operates from a single device is also within the scope of the present invention. In addition, while preferred embodiments are described in terms of access rights of a "user" (or equivalently, of a "requester"), this is for purposes of illustration and not of limitation: alternatively, access rights or permissions which are appropriate for a requester may be determined with reference to other entities. (For example, access rights may be associated with a role, with an authority level, or with a specific user identification, and so forth.) It should also be noted that the techniques of the present invention do not require changing the commercially-available LDAP directory implementation. –

Please replace the paragraph that begins on Page 17, line 8 and carries over to Page 19, line 2 with the following marked-up replacement paragraph:

-- The present invention uses an object model for software package installation, in which a framework is defined for creating one or more objects which comprise each software installation package. The present invention discloses a technique for using that object model to enable multiple versions of an installation package to be flexibly and efficiently assembled from multiple versions of its objects, according to different access rights of intended receivers of the installation package. The present invention also discloses optional authentication of the receivers prior to assembling an ~~installing~~ installation package for distribution thereto. These techniques will be described in more detail herein. Preferred embodiments of the software object model and

framework are described in the related inventions. As disclosed therein, each installation object preferably comprises object attributes and methods for the following:

1) A manifest, or list, of the files comprising the software package to be installed.

2) Information on how to access the files comprising the software package. This may involve:

a) explicit encapsulation of the files within the object, or

b) links that direct the installation process to the location of the files (which may optionally include a specification of any required access protocol, and of any compression or unwrapping techniques which must be used to access the files).

3) Default response values to be used as input for automatically responding to queries during customized installs, where the default values are preferably specified in a response file. The response file may specify information such as how the software package is to be subset when it is installed, where on the target computer it is to be installed, and other values to customize the behavior of the installation process.

4) Methods, usable by a systems administrator or other software installation personnel, for setting various response values or for altering various ones of the default response values to tailor a customized install.

5) Validation methods to ensure the correctness and internal consistency of a customization and/or of the response values otherwise provided during an installation.

6) Optionally, localizable strings (i.e. textual string values that may be translated, if desired, in order to present information to the installer in his preferred natural language).

7) Instructions (referred to herein as the "command line model") on how the

installation program is to be invoked, and preferably, how return code information or other information related to the success or failure of the installation process may be obtained.

8) The capabilities of the software package (e.g. the functions it provides).

9) A specification of the dependencies, including prerequisite or co-requisites, of the software package (such as the required operating system, including a particular level thereof; other software functions that must be present if this package is to be installed; software functions that cannot be present if this package is installed; etc.). --

Please replace the paragraph on Page 39, lines 8 - 15, with the following marked-up replacement paragraph:

-- Fig. 10 depicts a preferred embodiment of logic with which the installation time processing may be performed. This processing is described in terms of installation from a directory server on which one or more versions of the suite beans and component beans, as well as their objects, are stored (or are otherwise accessible), across a network to one or more target devices. It will be obvious to one of ordinary skill in the art how the process of Fig. 10 may be altered for use in other installation scenarios, including installation on a stand-alone machine which is not connected to a network, or a local installation where the client and server are co-resident. --

Please replace the paragraph on Page 41, lines 4 - 8 with the following marked-up replacement paragraph:

-- Upon receiving the Suite object, the client may then request (Block 1030) delivery of a

~~Machine~~ Group object. A ~~Machine~~ Group object contains one or more component objects which are appropriate to this particular type of client device, as previously described (and which may be specific to this requester's access rights). After receiving this request, the directory server returns the ~~Machine~~ Group object to the requester (Block 1035). –

Please replace the paragraph that begins on Page 41, line 18 and carries over to Page 42, line 3 with the following marked-up replacement paragraph:

– Upon receiving the .jar file, the client executes the pre-install program (Block 1055), if one has been defined. Block 1060 then executes the installation of the component itself, and Block 1065 ~~1070~~ executes the post-install program, if one has been defined for this component. (Refer to the description of Blocks 830 through 855, above, for more information on pre- and post-install programs.) --